Solutions

Marking: **M1** a correct method, **A1** the correct answer, **B1** an independent mark.

Each answer shows the steps, then the **result**.

2.1

$100^3 = 10^6$

3 levels

2.2

`city = 'Ningbo'`: yes

`city = 'Ningbo' AND age > 30`: yes, a range inside one city

`age > 30`: no, `age` is the second column

2.3

the index needs about one random page read per matching row

40% of the rows touch nearly every page, many times, so one sequential scan is cheaper

2.4

each insert also updates 5 index trees: 10^5 index updates a second

this slows writes and uses space and memory

keep only the indexes that frequent queries use, found from the query log and `EXPLAIN`

3.1

(a) $10^7/100 =$ **10^5** pages

(b) 3 index pages + 1 data page = **4**

(c) unclustered: about 10^5 random reads, one per row; clustered: the rows sit together, about $10^5/100 =$ **1000** pages

(d) when the matching rows equal the page count: $10^5/10^7 =$ **1**%; in practice less, as random reads are slower

3.2

(a) it reads every row of `orders` to find 12

(b) `CREATE INDEX idx_orders_customer ON orders (customer_id);`

(c) the index is sorted by `email`, not by `LOWER(email)`; `CREATE INDEX ON users (LOWER(email));`

3.3

(a) `(user_id, time)`: one user's events are together and sorted by time, so the range is one short scan; `(time, user_id)`: the range covers all users' events in that time, which must all be read

(b) $50\,000 \times 3 \times 4 =$ **6×10^5** writes a second

exercises.lol exercises.com.cn

(c) each month's partition and its indexes are small; queries skip months outside the range, and old months can be dropped at once

(d) one index `(user_id, time)` on monthly partitions, and no others unless a common query needs them: it serves the main query, while each extra index adds 2×10^5 writes a second